Jadavpur University

Department of Computer Science and Engineering

NETWORKS LAB ASSIGNMENT 5

BCSE UG-III

Student : Arjeesh Palai
Roll No. : 002310501086
Group : A3
Date : 10 / 11 / 2025

Assignment 5

Task: Packet Tracer & traffic analysis with Wireshark: capture traffic, inspect protocol stacks (Ethernet/IP/TCP/UDP/ICMP/HTTP), use filters, follow streams, view raw bytes, compute latencies, identify NIC OUIs, and generate protocol/flow statistics.

Deliverables: Screenshots and explanations for ICMP/ARP (or ICMPv6 ND), HTTP request/response timing, IP addresses, Host header hex, followed flows, server packet (TCP Src=80) with Ethernet details, NIC manufacturers & OUIs, protocol percentages, and a Flow Graph.

Answers

Q1. ICMP and initial address resolution (ARP / ICMPv6 ND)

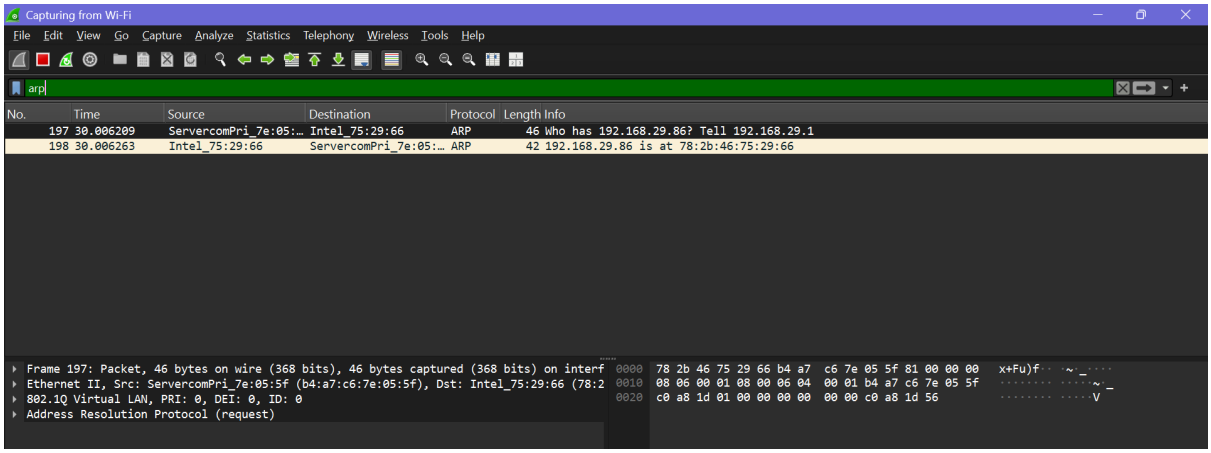

Figure 1: IPv4 ARP request/reply before ICMP echo.

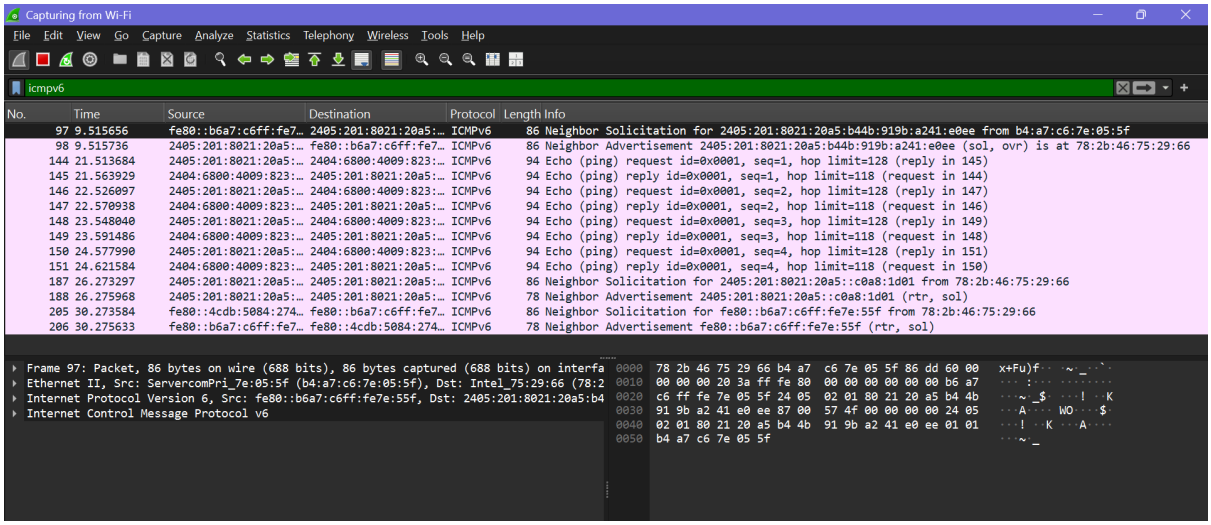

Figure 2: IPv6 Neighbor Solicitation/Advertisement before ICMPv6 echo.

Answer: Started a fresh capture and ran `ping`. L2 resolution precedes ICMP: ARP (IPv4) or ICMPv6 ND (IPv6). After resolution, Echo Request/Reply pairs appear for the ping exchange.

Q2. Web traffic inspection (protocols, RTT, addresses, Host)

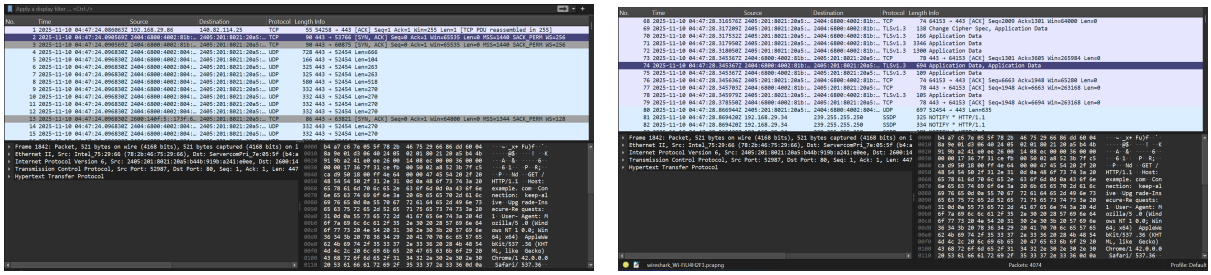

Figure 3: Unfiltered protocol glimpses (1/3).

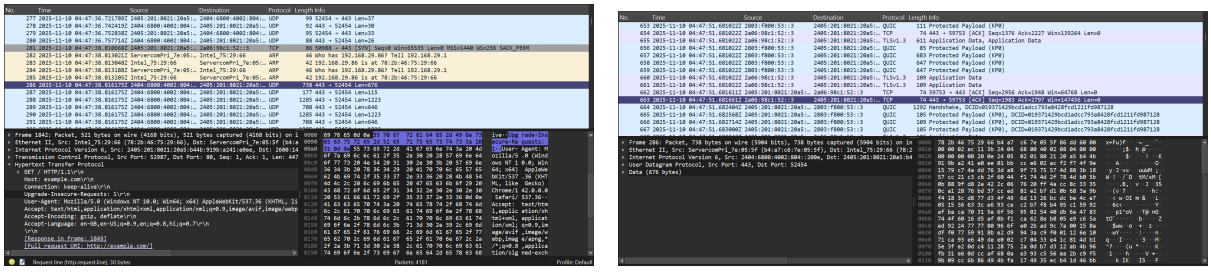

Figure 4: Unfiltered protocol glimpses (2/3).

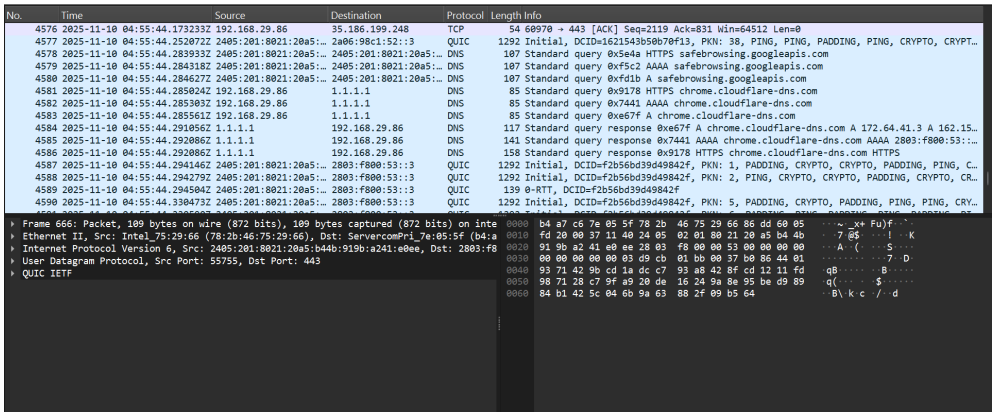

Figure 5: Unfiltered protocol glimpses (3/3).

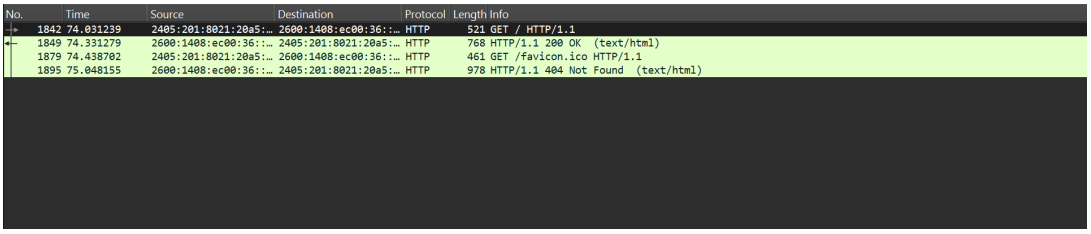

Figure 6: HTTP GET packet used for latency measurement.

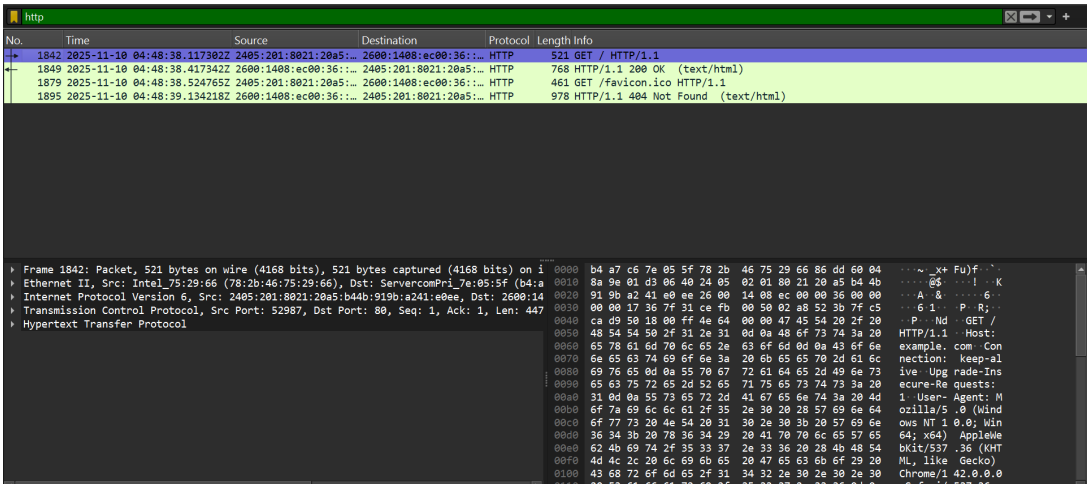

Figure 7: Matching HTTP 200 OK reply with timestamp.

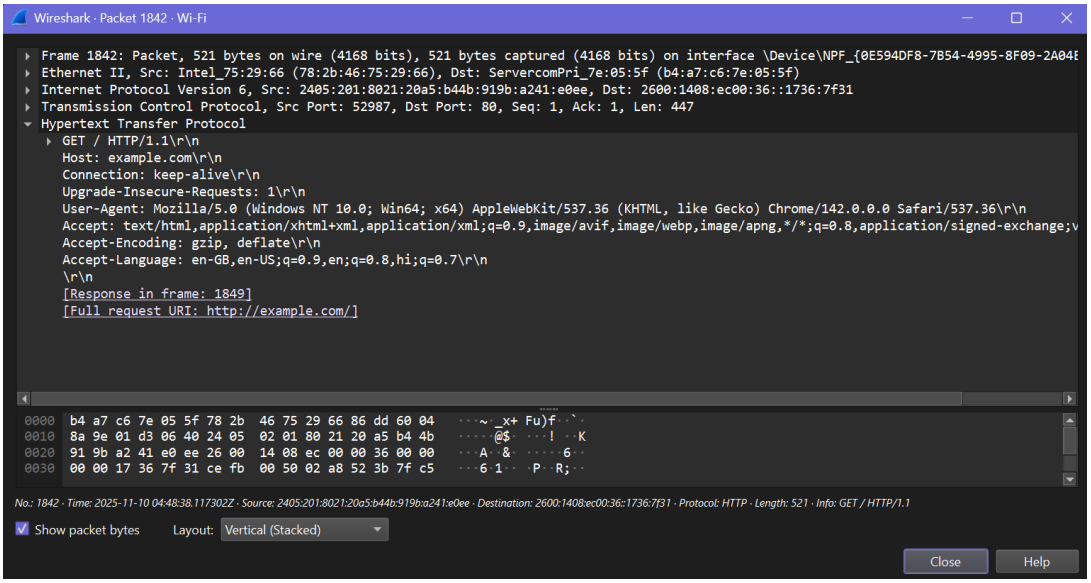

Figure 8: Source/Destination IPs from the GET packet.

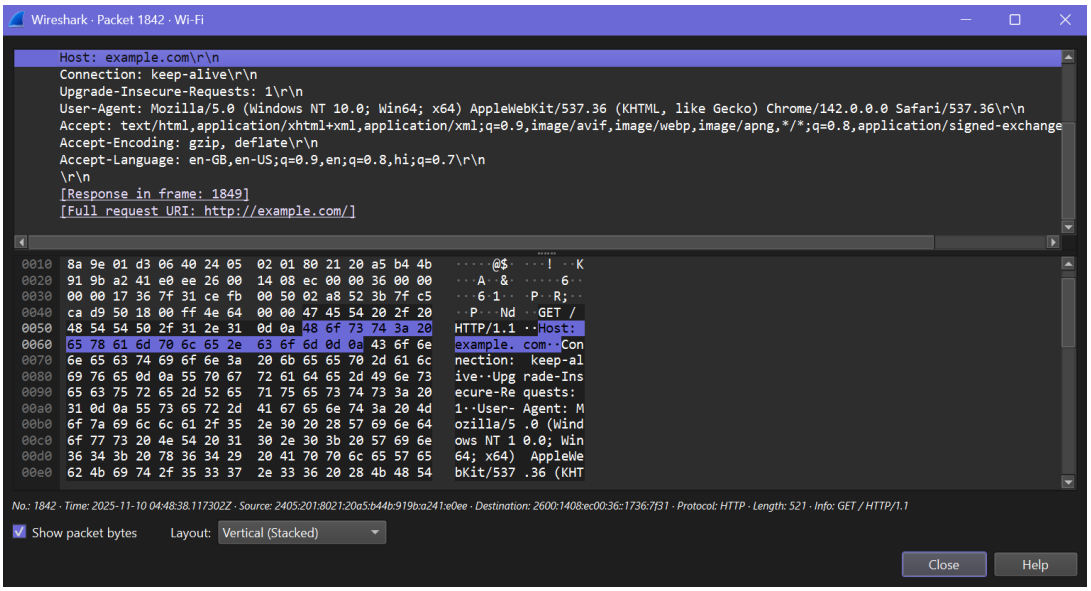

Figure 9: Packet Bytes and HTTP header showing `Host:`.

(a) Protocols: TCP, UDP, HTTP, ARP, DNS, TLS 1.2/1.3, QUIC, SSDP, ICMPv6.

(b) HTTP latency: GET at $t = 74.031239$ s; 200 OK at $t = 74.331279$ s $\Rightarrow \Delta t =$ **0.30004** s.

(c) IPs: Source (host) `2405:201:8021:20a5:b44b:919b:a241:e0ee`; Destination (site) `2600:1408:ec00:36::1736:7f31`.

(d/e) Host: `Host: example.com` in the HTTP layer.

Q3. Hex and ASCII in Packet Bytes

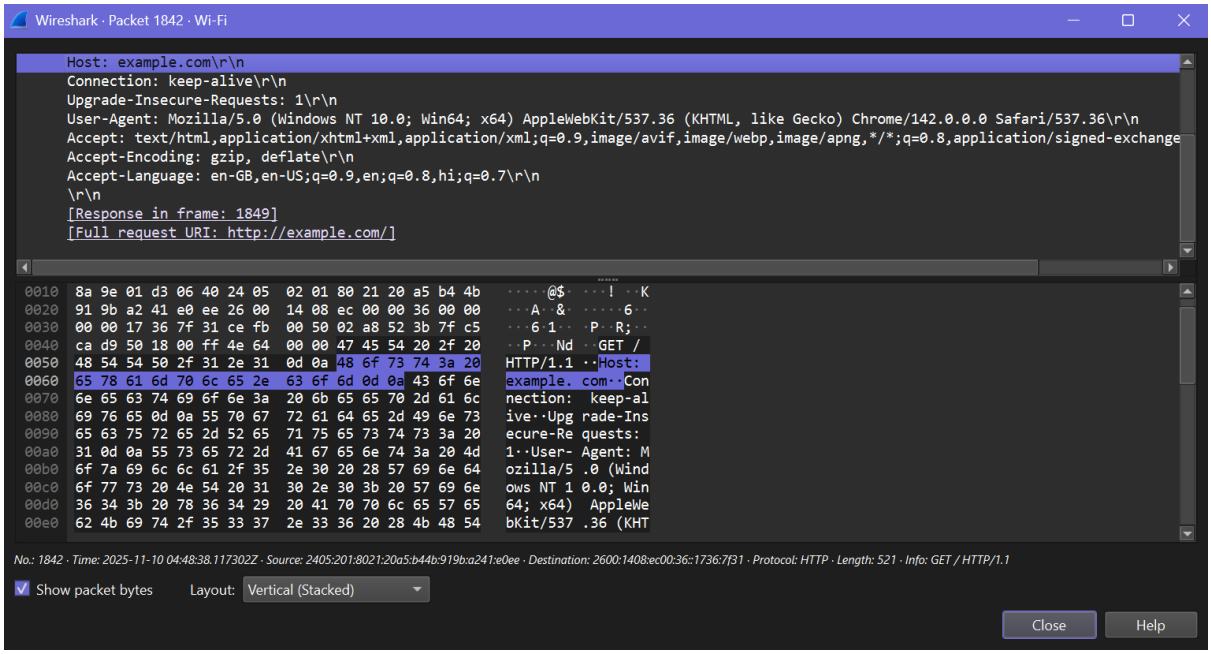

Figure 10: Packet Bytes pane showing aligned Hex and ASCII.

Answer: The lower pane displays offset, hex bytes, and ASCII; the `Host: example.com` substring is visible with matching hex.

Q4. First 4 bytes of Hex for the `Host` value

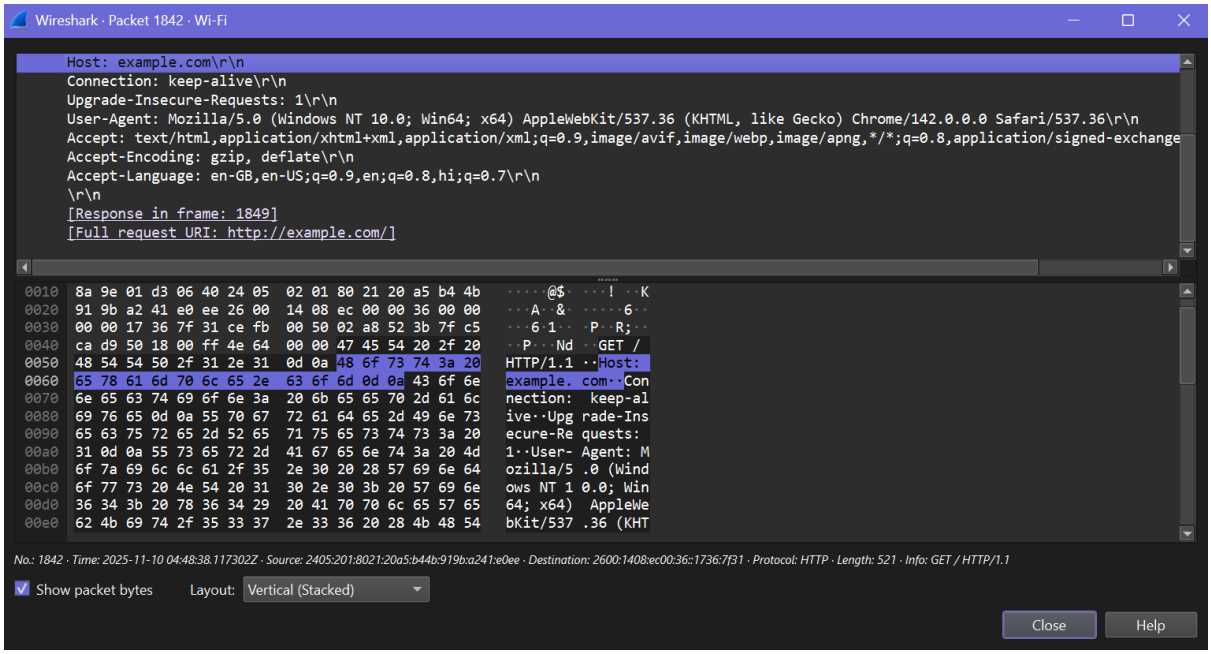

Figure 11: Locating the start of `example.com` for hex extraction.

Answer: The first four bytes of the value (*exam*) are **65 78 61 6d**.

Q5. Follow a conversation (flow)

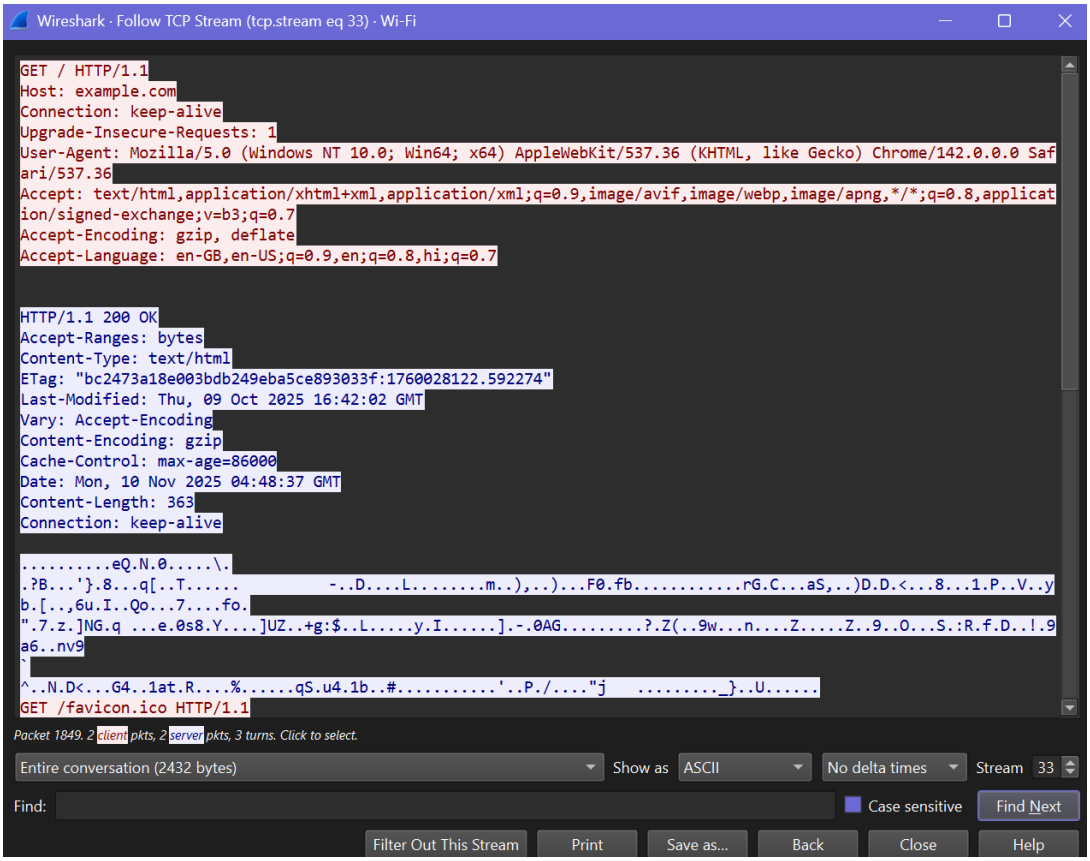

Figure 12: Follow → TCP Stream: reassembled request.

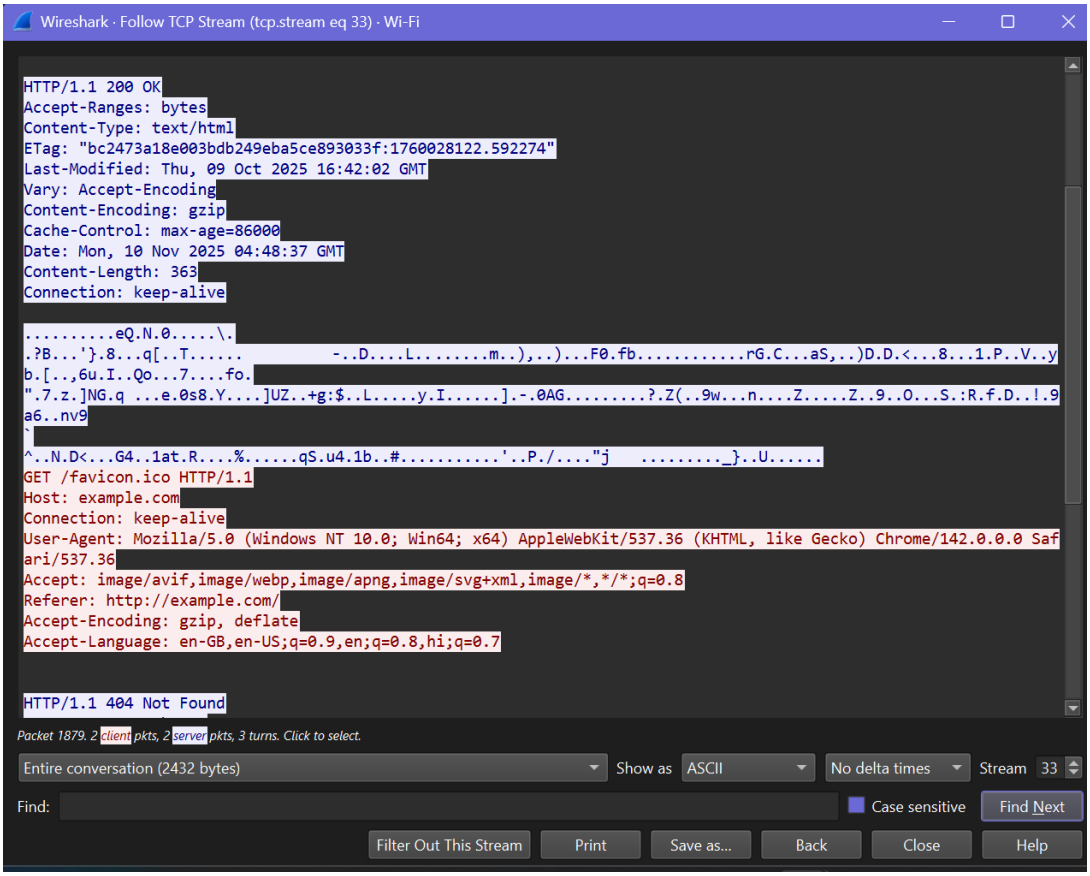

Figure 13: Follow → TCP Stream: reassembled response.

Answer: Using Follow TCP Stream reconstructs the HTTP GET and the subsequent 200 OK with payload.

Q6. HTTP packet from server (TCP Source Port = 80)

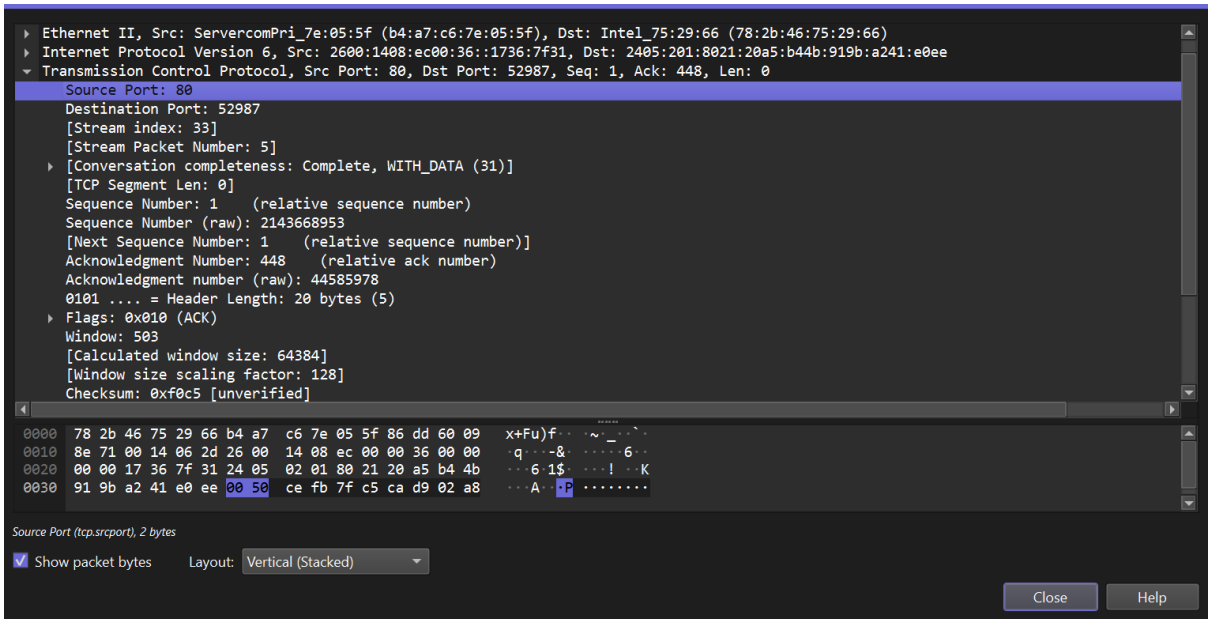

Figure 14: Server → client segment with TCP Src=80 and Ethernet expanded.

Answer: A server-to-client HTTP packet shows TCP Source Port = 80; expanding Ethernet reveals MAC addressing.

Q7. NIC manufacturers (vendor resolution)

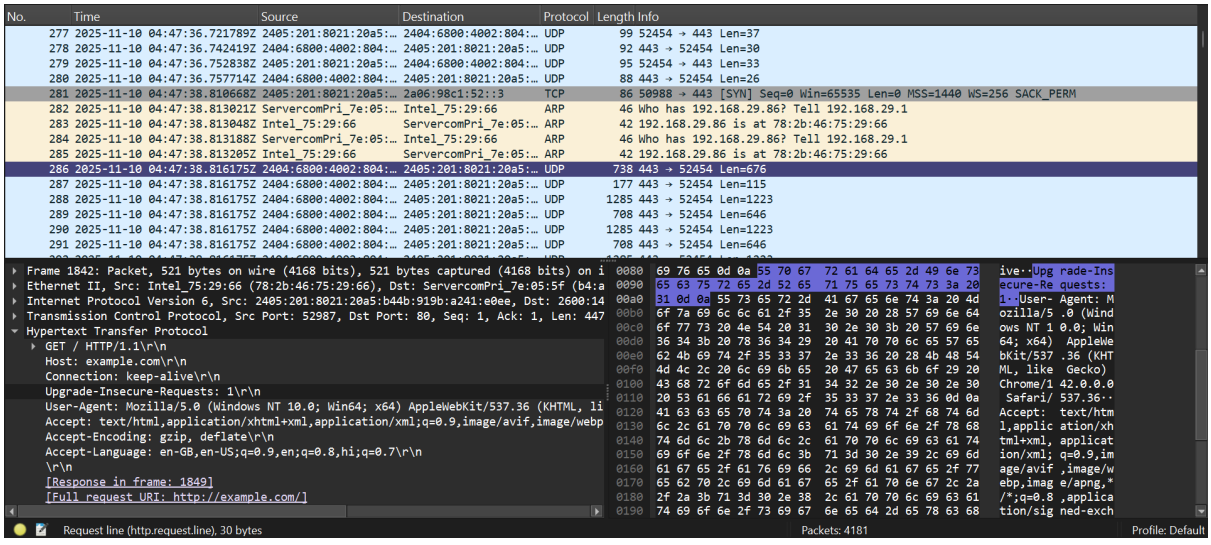

Figure 15: Ethernet II details with vendor names (name resolution enabled).

Answer: Vendor mapping indicates the host NIC as **Intel** and the server NIC as **ServercomPri** (example frames shown).

Q8. Hex OUIs (first three bytes of each MAC)

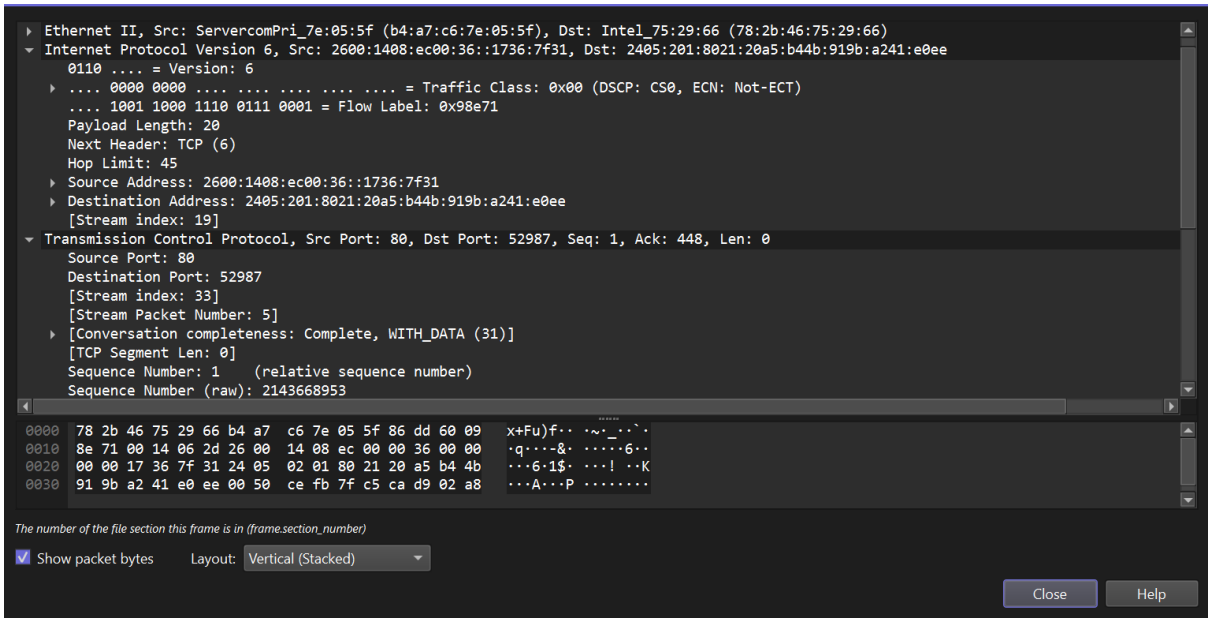

Figure 16: Reading OUIs (first 3 bytes) from MAC addresses.

Answer: OUIs observed: Intel **78** : **2b** : **46**; ServercomPri **b4** : **a7** : **c6**.

Q9. Protocol percentages (TCP vs UDP)

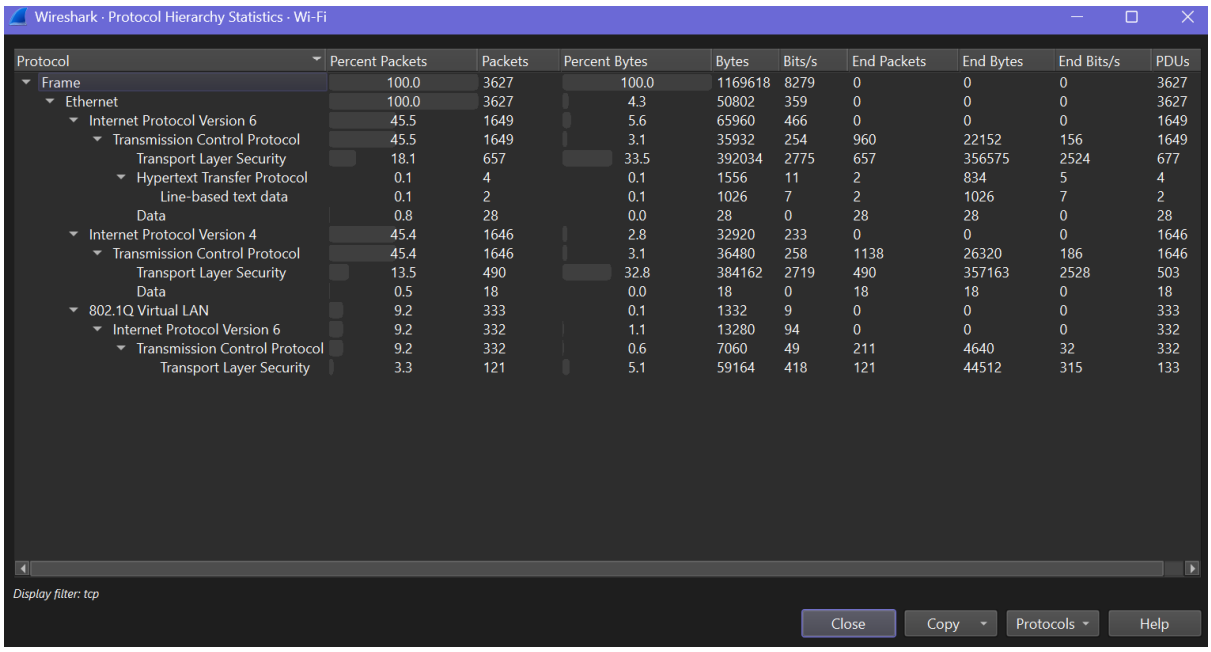

Figure 17: Protocol Hierarchy (*capture must be unfiltered*).

Answer: Clear display filters → **Statistics** → **Protocol Hierarchy** to obtain accurate TCP/UDP percentages for the entire capture.

Q10. Flow Graph

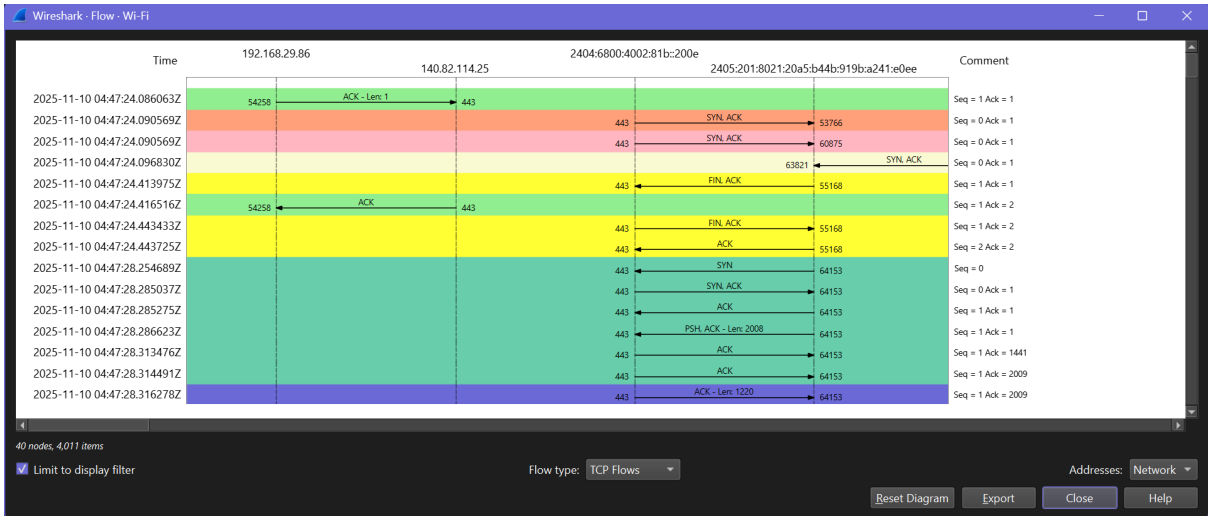

Figure 18: Statistics → Flow Graph summarizing per-flow timelines.

Answer: The graph shows SYN/SYN-ACK/ACK transitions and request/response ordering across concurrent connections.

Appendix: Capture Steps

- Selected the correct NIC and started capture.
- Enabled MAC name resolution for vendor IDs.
- Generated traffic: `ping`, web browsing (HTTP/HTTPS), DNS lookups.
- Used filters (`http`, `tcp`, `dns`), Follow Streams, Protocol Hierarchy, and Flow Graph.